ESP, A DISTRIBUTED ARCHITECTURE LSI MACHINE

Carver A. Mead

California Institute of Technology
Pasadena, California 91109

Summary

The Externally Sequenced Processor (ESP) is a system which embodies a complete separation of the control and data processing functions of the machine. The ESP is organized so that additional functional capabilities as well as peripherals can be added. The interfacing requirements are particularly straightforward.

Introduction

Any operating digital system (including its resident software) can be conceptually separated into two distinct sctions; one associated with the control of the system, and the other primarily dedicated to data storage and processing. Although traditional processors do not take advantage of this distinction, a great deal of optimization in the hardware can be achieved by partitioning a system along these lines. Further, each of these major machine subdivisions contains two quite distinct types of binary numbers. The control section contains instructions and instruction locations, while the data storage and processing section contains data words and data memory addresses. Each of these four quantities is treated in quite a different way in the execution of typical system functions and once again, partitioning the system so that these quantities can be handled by separate pieces of hardware allows considerable optimization of machine performance without pushing the technology for ultra-fast cycle times.

The ESP system departs from the long tradition of stored program machines by recognizing that most, if not all of the software will be resident in some type of read-only medium. Rather than a mere read-only memory, we have chosen as our medium a self-contained programmable sequencer-controller which contains not only the instructions, but the entire program sequence as well.

System Organization

A block diagram of the ESP is shown in Fig. 1. The control section is composed of as many control modules as is necessary to contain the amount of program resident in the system. However, the modules are not merely read-only memories in which program is stored, but also contains all of the sequencing logic and capabilities for looping, branching, etc. With more than one control module, fully independent subroutine capabilities and automatic nesting are inherent in the system. One and only one control module has complete control of the system at any time. It transmits and receives information on two buses, the command bus (used exclusively for instructions) and the data bus (primarily used for data transfers between members of the data section of the machine). The system operates in a completely synchronous manner with two-phase clock being supplied to all modules of the system.

The primary members of the data section of the machine are the data memory module and various classes of arithmetic-logic modules. The most important data centered operation in the machine in most installations is that of maintaining a data memory. This operation is carried out by the data memory control module which interfaces the actual memory chips comprising the data memory to the bus system. Several arithmetic and logic modules may also be attached to the system to do the conventional classes of operations upon the data appearing on the data bus under the control of the command bus. Finally, any number of periph-

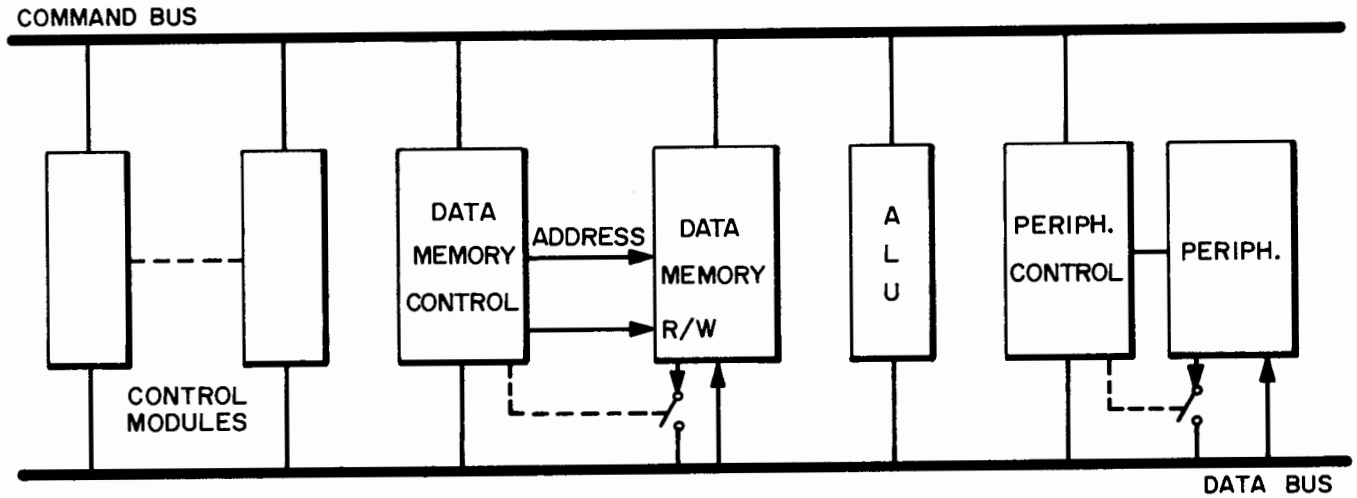

FIG. 1 ESP BLOCK DIAGRAM

erals can be attached to the bus in the same manner that the data memory was. A peripheral control module looks at the command and data buses along with outputs from the peripheral, and provides control signals necessary for operating the peripheral and allowing the peripheral to either receive or send data on the data bus. Each of the modules in the data section has its own instruction set, recognizing instructions on the command bus and has access to data on the data bus. With the exception of certain peripherals, all the data modules execute one instruction in one machine cycle time. Hence, all of their operations are completely under the control of the control modules at all times. By careful design of the data modules it was not necessary to compromise the instruction set to execute instructions in one cycle time, primarily because the operations done by each module are very similar. Conflicts were resolved by partitioning rather than by time sequencing. This, in fact, may be viewed as the central philosophy of the entire machine.

A sequence is initiated with the control module placing an instruction upon the command bus. This instruction is decoded by the appropriate data module, executed, and the result placed on the data bus during the following clock cycle. By this time the control module will have placed the next instruction on the command bus and that instruction will be executed upon the data which will then be present on the data bus. As a simple example, let us consider the transfer of data from a location in memory to the arithmetic logic unit. During the first clock cycle, the control module places a read memory command on the command bus. This instruction is decoded in the data memory control module and during the second clock cycle the contents of the currently addressed memory location will appear on the data bus. At the same time the control module will place a store command on the command bus and the arithmetic logic unitwill take the contents of both the command and data buses during the second clock cycle, interpret its instruction, and store the contents of the data bus in one of its registers. This "source-destination" doublet is characteristic of a great deal of the action which takes place on the buses.

The Control Module

Given a controller with the appropriate characteristics, this machine possess great generality, flexibility and is indefinitely extendable. It is clear from the above argument that the most crucial requirement of the system is for a controller of rather unique characteristics. It should be able to contain large quantities of program without consuming a large amount of chip area; it should contain the capability of branching on all sorts of conditions, and sequencing through program states in a completely general way. It is not obvious that such a controller is easily implementable or even possible in the LSI medium, but in fact, it turns out to be. The controller used is a programmable logic array, which has been described elsewhere.[1] The device is shown schematically in Fig. 2. It consists

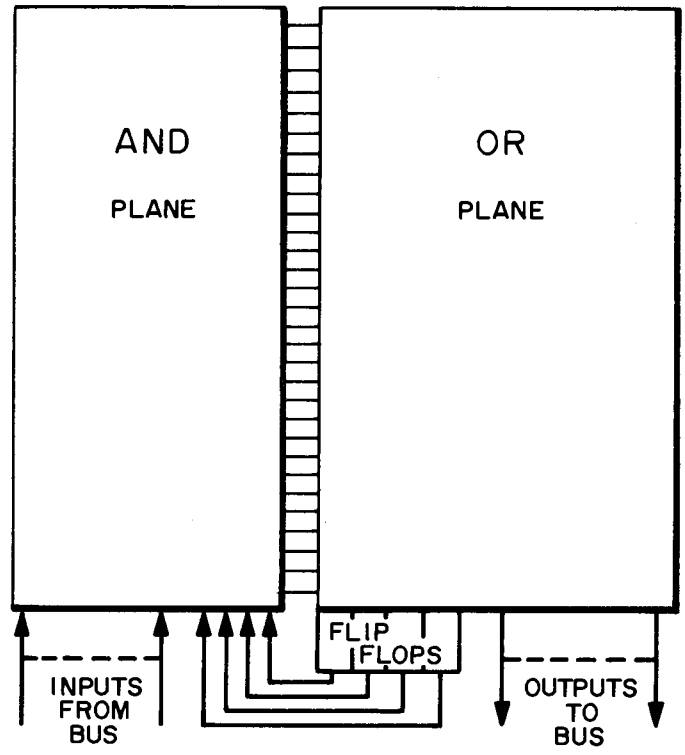

FIG. 2 PROGRAMMABLE LOGIC ARRAY

of a set of input lines running vertically into an AND matrix whose outputs are horizontal lines which can be programmed to decode any AND combination of input terms (or of any subset of the input terms). The horizontal outputs of the AND matrix form inputs to a OR matrix whose outputs run vertically out to the buses and also to a number of flip-flops whose outputs are fed back into the AND matrix. It is clear that this device is simply a straightforward embodement of one of the two canonical representations of any logic function and therefore can provide any arbitrary function of the desired inputs. However, the real power of the device manifests itself in rather non-obvious ways. Since the inputs to the flip-flops can be any arbitrary logic function of the external inputs and the present contents of the flip-flops; the next flip-flop state can be programmed to be any arbitrary function of the inputs and the past history of the machine. This, of course, is just the definition of the completely general sequencial controller. Since the program is placed in the machine in the form of contacts being made or not made to transistors in the AND and OR arrays the AND plane can be programmed to recognize any input variable, its compleement or neither. Hence, the AND plane represents a content addressable program store with complete flexibility to ignore variables not of interest. In practice, the flip-flops

contain the internal state of the controller (the program step) and are sequenced on each clock cycle, depending upon the current machine state and the external inputs. Each horizontal line in the sequencer corresponds to one line of machine code. Jumps are executed merely by loading a new flip-flop state dependent upon the current state and the inputs. Since more than one horizontal line may become activated during a clock cycle, (depending upon the state of the inputs) it is possible to write tight inner-loops of one or two instructions which contain N-way branches, the branches in fact, not requiring any time for execution. This capability illustrates perhaps the most powerful feature of the machine, that is separate instructions are not required for branches and thus the total control operation can proceed within the control module and never be seen by the balance of the machine. Since in most machine installations something like 90% of the machine cycles are used for control operations, a great increase in machine performance is achieved by this partitioning without engaging in the "NANO-second race." The current machine structure employs a 10 bit command bus and an 8 bit data bus, the control modules are implemented in silicon gate technology (either P or N channel) the chip size is 80 x 160 mils and occupies a 22 pin package.

Data Memory Control

The data memory control module contains 8, 16 bit memory address registers, only one of which is selected at any given time. The outputs of the selected register go directly to the memory address drivers. Either the high order or the low order half of the selected register can be loaded from the data bus or can transmit its contents to the data bus. Included in the module is a 16 bit adder which allows base-displacement type addressing with any relative address within the limits ± 128. Indexing is achieved by latching the index in a data bus latch and executing a read and increment (or write and increment) operation which places the contents of the currently addressed memory location on the data bus while incrementing the memory address by the contents of the index latch (and similarly for the write instructions). Great flexibility in addressing is thus achieved with a very simple instruction set. The data memory control module has been implemented in silicon gate chip 115 x 120 mils in size and occupies a 40 pin package. It also, of course, controls the read-write memory control and an outboard driver which allows the memory to throw its contents to the data bus, signified by the switch in Fig. 1.

Arithmetic and Logic Operations

Two arithmetic-logic modules have been designed for the machine. The first is a rather standard ALU containing 7 registers and allowing any of the standard uniary or binary operations to be executed between the accumulator and the other registers. The results may be returned to either the designated register or to the accumulator. The data bus is treated as the 8th register. This unit in silicon gate technology occupies a chip 100 x 100 mils and a 22 pin package. In addition, a hardware multiply module has been designed which allows great flexibility in either single or multiple precision multiplications of either magnitude numbers or twos complement signed numbers in all combinations and has a built-in adder and accumulator so that it can accumulate the sum of products without resorting to the arithmetic logic unit. The high or the low order part of the product can be placed on the data bus on command or the entire product can be retained in the accumulator for later use.

Acknowledgement

The machine was in large measure, a class project in a graduate course in LSI Design. The individuals primarily responsible for the various portions of the system were: PLA Controller - John Gord, Avi Gover, Jim Atherton, Vincent Wong; Data Memory Control - Mike Wimbrow, Bob Offerman, Steve Bissett, Ed Cheng; ALU - Paul Schluter, Joe Elmers, Vincent Wong; Multiplier - Rodney Matsumoto, Hon Hing So; and Software - Steve Colley.

The system definition was the product of much discussion with Steve Colley and Kent Gordon. The work was supported in part of the Office of Naval Research (D. Ferry).

Reference

1. Texas Instruments Electronics Series, MOS/LSI Design and Application, R.E. Sawyer and J.R. Miller, Editors. McGraw-Hill Book Company, Chapter 8, p. 229 (1972).